Some Proof Techniques for Language Analysis

Cormac Flanagan
University of California, Santa Cruz

Induction

- Probably the single most important technique for the study of formal semantics and type systems of programming languages.

- Of several kinds
  - mathematical induction (the simplest)
  - well-founded induction (the most general)
  - structural induction (the most widely used in this context)

Mathematical Induction

- Goal: prove that $\forall n \in \mathbb{N}. P(n)$

- Strategy: (2 steps)
  1. Base case: prove that $P(0)$
  2. Inductive case:
     - pick an arbitrary $n \in \mathbb{N}$
     - assume that $P(n)$ holds
     - prove that $P(n+1)$
     - or, formally, prove that $\forall n \in \mathbb{N}. P(n) \Rightarrow P(n+1)$

Mathematical Induction: Notes

- The inductive step looks similar to the goal but it is simpler because of the assumption that $P(n)$ holds.

  $\forall n \in \mathbb{N}. P(n-1) \Rightarrow P(n)$ vs. $\forall n \in \mathbb{N}. P(n)$

- Why does mathematical induction work?
  - The key property of $\mathbb{N}$ is that there are no infinite descending chains of naturals.
  - For each $n$, $P(n)$ can be obtained from the base case and $n$ uses of the inductive case.

Example of Mathematical Induction

- Recall the evaluation rules for IMP commands.
- Prove that if $\alpha(x) \leq 6$ then $<\text{while } x \leq 5 \text{ do } x := x + 1, \sigma_0> \Downarrow \sigma_0(x := 6)$$<x, \sigma_0> \Downarrow 6 <5, \sigma_0> \Downarrow \text{false}$
- Reformulate the claim:
  - Let $W = \text{while } x \leq 5 \text{ do } x := x + 1$
  - Let $\alpha_i = \sigma_i(x := 6 - i)$
  - Claim: $\forall i \in \mathbb{N}. <W, \alpha_i> \Downarrow \alpha_0$
  - Now the claim looks provable by mathematical induction on $i$.

Example of Mathematical Induction (Base Case)

- Base case: $i = 0$ or $<W, \alpha_0> \Downarrow \alpha_0$
  - To prove an evaluation judgment, construct a derivation tree:

  $\alpha_0(x) = 6$

  $<x, \alpha_0> \Downarrow 6 <5, \alpha_0> \Downarrow \text{false}$

  $<\text{while } x \leq 5 \text{ do } x := x + 1, \alpha_0> \Downarrow \alpha_0$
  - This completes the base case.
Example of Mathematical Induction (Inductive Case)

- We must prove \( \forall i \in \mathbb{N}. <W, \sigma_i> \Downarrow \sigma_0 \Rightarrow <W, \sigma_{i+1}> \Downarrow \sigma_0 \)
- The beginning of the proof is straightforward:
  - Pick an arbitrary \( i \in \mathbb{N} \)
  - Assume that \( <W, \sigma_i> \Downarrow \sigma_0 \)
  - Now prove that \( <W, \sigma_{i+1}> \Downarrow \sigma_0 \)
  - We must construct a derivation tree:

\[
\begin{align*}
\sigma_0 & \Downarrow 5 - i \\
\sigma_i & \Downarrow \text{true} \\
\sigma_0 & \Downarrow <x:=x+1; W, \sigma_i> \\
\sigma_0 & \Downarrow <\text{while } x \leq 5 \text{ do } x := x + 1, \sigma_i> \\
\sigma_0 & \Downarrow <W, \sigma_i> \\
\end{align*}
\]

Discussion

- A proof is more powerful than running the code and observing the result. (Why?)
- We proved termination and correctness. This combination is called total correctness.
- Mathematical induction is good when we prove properties of natural numbers.
  - But in programming-language analysis we most often prove properties of expressions, commands, programs, input data, etc.
  - We typically need a more powerful induction principle.

Well-Founded Induction

- A relation \( < \subseteq A \times A \) is well-founded if there are no infinite descending chains in \( A \).
  - Example: \( \preceq \) (\( (x, x + 1) | x \in \mathbb{N} \))
  - the predecessor relation
  - Example: \( < \) (\( (x, y) | x, y \in \mathbb{N} \) and \( x < y \))
- Well-founded induction:
  - To prove \( \forall x \in A. P(x) \) it is enough to prove \( \forall x \in A. (\forall y \preceq x \Rightarrow P(y)) \Rightarrow P(x) \)

- If \( < \) is \( \preceq \), then we obtain a special case of mathematical induction.

Well-Founded Induction: Examples

- Consider \( \preceq \subseteq \mathbb{N} \times \mathbb{N} \) with \( x \preceq y \) if \( x + 2 \geq y \)
  - \( \forall x \in \mathbb{N}. (\forall y < x \Rightarrow P(y)) \Rightarrow P(x) \) is equivalent to \( P(0) \land P(1) \land \forall n \in \mathbb{N}. (P(n) \Rightarrow P(n + 2)) \)
- Consider \( \preceq \subseteq \mathbb{Z} \times \mathbb{Z} \) with \( x \preceq y \) if \( y > 0 \) and \( x = y \cdot x - 1 \) or \( y > 0 \) and \( y = x + 1 \)
  - \( \forall x \in \mathbb{Z}. (\forall y < x \Rightarrow P(y)) \Rightarrow P(x) \) is equivalent to \( P(0) \land \forall x \leq 0, P(x) \land P(x + 1) \land \forall x > 0, P(x) \Rightarrow P(x + 1) \)

Well-Founded Induction: Examples (cont.)

- Consider \( \preceq \subseteq (\mathbb{N} \times \mathbb{N}) \times (\mathbb{N} \times \mathbb{N}) \) and \( (x_1, y_1) \preceq (x_2, y_2) \) if \( x_1 = x_2 + 1 \lor (x_1 = x_2 \land y_2 = y_1 + 1) \)
  - This leads to the induction principle \( P(0,0) \land \forall x,y. P(x,y) \Rightarrow P(x+1,y) \land P(x,y+1) \)
  - This is sometimes called lexicographic induction.
**Structural Induction**

- Recall $e ::= n \mid e_1 + e_2 \mid e_1 \ast e_2 \mid x$
- Define $\prec \subseteq \exp \ast \exp$ such that
  - $e_1 \prec e_2$ if $e_1 = e_2$
  - $e_1 \prec e_2$ if $e_1 \prec e_2$ and $e_2 \prec e_2$
- and no other elements of $\exp \ast \exp$ are related by $\prec$
- To prove $\forall e \in \exp. P(e)$
  1. Prove $\forall n \in \mathbb{Z}. P(n)$
  2. Prove $\forall x \in L. P(x)$
  3. Prove $\forall e_1, e_2 \in \exp. P(e_1) \land P(e_2) \Rightarrow P(e_1 + e_2)$
  4. Prove $\forall e_1, e_2 \in \exp. P(e_1) \land P(e_2) \Rightarrow P(e_1 \ast e_2)$

**Example of Induction on Structure of Expressions**

- Let
  - $L(e)$ be the number of literals and variable occurrences in $e$
  - $O(e)$ be the number of operators in $e$
- Prove that $\forall e \in \exp. L(e) = O(e) + 1$
- By induction on the structure of $e$
  - Case $e = n$. $L(e) = 1$ and $O(e) = 0$
  - Case $e = x$. $L(e) = 1$ and $O(e) = 0$
  - Case $e = e_1 + e_2$.
    - $L(e) = L(e_1) + L(e_2)$ and $O(e) = O(e_1) + O(e_2) + 1$
    - By induction hypothesis $L(e_1) = O(e_1) + 1$ and $L(e_2) = O(e_2) + 1$
    - Thus $L(e) = O(e) + 1$
  - Case $e = e_1 \ast e_2$. Same as the case for $+$.

**Another Proof**

- Prove that IMP is deterministic
  - $\forall e \in \exp. \forall \sigma \in \Sigma. \forall n, n' \in \mathbb{N}. \langle e, \sigma \rangle \Downarrow n \land \langle e, \sigma \rangle \Downarrow n' \Rightarrow n = n'$
  - $\forall b \in \exp. \forall \sigma \in \Sigma. \forall t, t' \in \mathbb{B}. \langle b, \sigma \rangle \Downarrow t \land \langle b, \sigma \rangle \Downarrow t' \Rightarrow t = t'$
  - $\forall c \in \text{Comm}. \forall \sigma, \sigma' \in \Sigma. \langle c, \sigma \rangle \Downarrow \sigma' \Rightarrow \sigma = \sigma'$
- No immediate way to use mathematical induction
- For commands, we cannot use induction on the structure of the command
- Consider the rule for while. Its evaluation does not depend only on the evaluation of its strict subexpressions

**Induction on the Structure of Derivations**

- Key idea: The hypothesis gives not only a $c \in \text{Comm}$ but also the existence of a derivation of $\langle c, \sigma \rangle \Downarrow \sigma'$.
- Derivation trees are also defined inductively, just like expression trees.
- A derivation is built of subderivations:

  $\langle x, \sigma, i \rangle \Downarrow 5 - i \Downarrow x \ast i, \sigma, i \Downarrow 6 - i$

  $\langle x \ast i, \sigma, i \rangle \Downarrow a_1$

  $\langle W, a_1 \Downarrow a_2 \rangle$

  $\langle W, a_1 \Downarrow a_2 \rangle$

  $\langle W, a_1 \Downarrow a_2 \rangle$

- While $x \ast 5$ do $x \Rightarrow x \ast 1, \sigma, i \Downarrow a_0$
- We adapt the structural induction principle to work on the structure of derivations.